Barış Ayyıldız 1990422252

1) $T(n) = T\left(\frac{n}{2}\right) + 1$

$T\left(\frac{n}{2}\right) = T\left(\frac{n}{4}\right) + 1$

$\vdots$

$T(1) = 1$

$\boxed{k = \log_2 n}$

$T(n) = T\left(\frac{n}{4}\right) + 1 + 1$

$T(n) = T(n/8) + 1 + 1 + 1$

$\vdots$

$T(n) = T(n/2^k) + \underbrace{1 + 1 + \cdots + 1}_{k}$

$T(n) = \log_2 n$

$T(n) \in \Theta(\log_2 n)$ //

2) $T(n) = 2T(n/2) + n$

$T(n) = 2\left(2T(n/4) + n/2\right) + n$

$T(n) = 4T\left(\frac{n}{4}\right) + 2n$

$T(n) = 4\left(2T\left(\frac{n}{8}\right) + \frac{n}{4}\right) + 2n$

$T(n) = 8T\left(\frac{n}{8}\right) + 3n$

$\vdots$

$T(n) = 2^k T\underbrace{\left(\frac{n}{2^k}\right)}_{1} + kn$

k times

$\boxed{k = \log_2 n}$

$T(n) = n + n\log_2 n \in \Theta(n\log_2 n)$ //

3) Best Case: Best case occurs when index is equal to k minus 1. If we can find that index after the first partition, best case happens.

$T_{best} = n \in \Theta(n)$

$\downarrow$

comes from the partition function

Worst case: Worst case happens when partition function cannot evenly divide the array by 2.

$T_w(n) = T(n-1) + \underbrace{cn}_{\text{partition}}$

$T_w(n-1) = T(n-2) + c(n-1)$

$T_w(n-2) = T(n-3) + c(n-2)$

$\vdots$

$T_w(1) = c$

$+$

---

$T_w(n) = T(n-2) + c(n-1) + cn$

$T_w(n) = T(n-3) + c(n-2) + c(n-1) + cn$

$\vdots$

$T_w(n) = c(n-k) + \dots + c(n-2) + c(n-1) + cn$

$= c(1+2+\dots+n-1) = \frac{c(n-1)\cdot n}{2} = \frac{cn^2-n}{2}$

$T(n) = \frac{cn^2-n}{2} \in \Theta(n^2)$

4) I used the same merge-sort algorith that I used in the question -2 . I just calculated the reverse ordered pairs while merging two arrays in constant time.

So the algorithm complexity is still $\Rightarrow T(n) \in \Theta(n \log_2 n)$

5) a) Brute Force Approach :

$$\left.\begin{aligned} T(n) &= T(n-1) + c \\ T(n-1) &= T(n-2) + c \\ &\vdots \\ T(1) &= 0 \end{aligned}\right\} k \text{ steps} \quad T(n) = T(1) + ck$$

$\boxed{k = n}$

$T(n) = 0 + cn$

$T(n) = n \in \Theta(n)$